VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

Dokumentácia k projektu do predmetu ISA

Nástroje monitorující a generující zprávy jednoduchých distance-vector protokolů

19. novembra 2018

Anton Firc (xfirca00)

Obsah

1 Distance-vector smerovacie protokoly

1.1 Popis

Distance-vector smerovacie protokoly používajú pre výpočet najlepšej cesty vzdialenosť do cieľovej siete. Vzdialenosť je počítaná podľa počtu smerovačov cez ktoré je potreba preniesť packet. Niektoré protokoly takisto rátajú s latenciou a inými faktormi ovplyvňujúcimi sieťovú prevádzku `(traffic)`. Pre výpočet najlepšej cesty je nutné aby si susedné smerovače pravidelne vymieňali informácie obsiahnuté v smerovacích tabuľkách. V smerovacích tabuľkách sú obsiahuté vektory vzdialeností do ostatných uzlov v sieti.

1.2 Smerovací protokol RIP

RIP(Routing Information Protocol) je jeden z najstarších smerovacích protokolov používajúci ako jedinú metriku počet skokov(hop count). Implementáciou maximálneho počtu skokov na ceste zo zdroja do cieľa zabraňuje vytváraniu smerovacích slučiek(routing loops). Pre výmenu informácií používa protokol `UDP` a rezervovaný port číslo `520`. RIP používa dva druhy správ - Request(žiada o zaslanie smerovacích informácii od susedných smerovačov) a Response(odpoveďna request, obsah smerovacej tabuľky). V dnešnej dobe však upadol do pozadia kvôli dobe konvergencie a škálovateľnosti(scalability) ktoré sú v porovnání s inými smerovacími protokolmi(EIGRP, OSPF...) slabé. Na druhú stranu je RIP veľmi jednoducho nakonfigurovateľný.

RIPv1

Po štarte smerovača a potom každých 30 sekúnd odosiela broadcastovú žiadosť cez každé rozhranie povolené pre použitie s RIPv1. Susedné smerovače obdržia žiadosť a ako odpoveď pošlú segment obsahujúci ich smerovaciu tabuľku. Žiadajúci smerovač potom aktualizuje svoju smerovaciu tabuľku IP adresou dosiahnuteľnej siete, počtom skokov a next-hop IP adresu alebo rozhranie z ktorého obdržal odpoveď. Aktualizácia však prebieha iba pokiaľ smerovacia tabuľka ešte neobsahuje danú sieť, alebo je sieť dostupná s nižším počtom skokov. Pokiaľ smerovač obdrží viac ciest do rovnakej siete s rovnakým počtom skokov, uloží všetky tieto cesty a informácie používa k rozdeľovaniu záťaže(load balancing). RIPv1 nemá podporu pre autentifikáciu smerovačov, je náchylný k rôznym útokom.

RIPv2

RIPv2 je vylepšením predchádzajúceho RIPv1. Podporuje prenos masky siete. Pre odľahčenie zariadení nepodieľajúcich sa na smerovaní používa pre výmenu informácií multicast. RIPv2 podporuje autentifikáciu smerovačov, buď vo forme textového hesla alebo MD5 hashovania. Pridaný bol aj router tag, ktorý napomáha odlíšiť cesty naučené protokolom RIP a inými protokolmi. Je spätne kompatibilný s verziou RIPv1.

RIPng

RIPng(RIP new generation) je rozšírenie RIPv2 pre podporu IPv6. Hlavnými rozdielmi je podpora IPv6 smerovania, používa IPSec autentifikáciu, využíva špeciálne kódovanie next-hop adresy pre skupinu ciest a používa port čislo `521`.

2 Návrh aplikácie

2.1 Sniffer RIP správ

Aplikácia odchytáva sieťovú premávku na zadanom sieťovom rozhraní a filtruje všetky správy smerovacieho protokolu RIPv1/v2/ng. Každá zachytená správa je spracovaná a zobrazená užívateľovi v štrukturovanej forme. Výpis a štruktúra informácií sú inšpirované programom Wireshark[1].

2.2 Podvrhávač falošných RIPng Response správ

Aplikácia slúži k šíreniu vlastných falošných RIPng Response správ. Správa je generovaná podľa parametrov zadaných užívateľom pri spustení. Po spustení dochádza k spracovaniu užívateľom zadaných informácií, vytvoreniu falošného packetu a potom zaslaniu na užívateľom zadané sieťové rozhranie.

3 Implementácia

3.1 Sniffer RIP správ

Pre implementáciu je použitý jazk C. Ihneď po spustení prebehne kontrola počtu a správneho formátu argumentov. Po úspešnej kontrole argumentov dojde k otvoreniu užívateľom zadaného rozhrania. Pre sledovanie sieťovej komunikácie sú využité funkcie z knižnice libpcap. Kódy pre implementáciu sledovania sieťovej komunikácie sú prevzaté z aplikácie sniffex.c[1]. Po úspešnom otvorení rozhrania je nastavený filter ktorý zachytáva iba `UDP` komunikáciu na portoch `520` a `521`. Odchytávanie a filtrovanie komunikácie zabezpečuje funkcia `pcap_loop()`. Odchytávanie packetov prebieha v nekonečnej slučke a končí sa až po obdržaní prerušenia `SIGINT`.

callback()

Funkcia callback je volaná funkcoiu `pcap_loop()` pre spracovanie každého jedného zachyteného packetu. Pri spracovaní je vypísané poradové číslo, zdrojová a cieľová MAC adresa a identifikovaná IP verzia packetu. Podľa typu IP adresy pokračujú vo spracovaní packetu funkcie `process_ipv4()` pre IPv4 alebo `process_ipv6()` pre IPv6 ktoré dostanú ako parameter ukazateľ na začiatok packetu. Funkcie získvajú jednotlivé časti packetu pomocou offsetu.

process_ipv4()

Funkcia začína s kontrolou dĺžky IP hlavičky, pokiaľ je správna tak je vypísaná verzia IP adresy a zdrojová a cieľová IP adresa. Následne je získaná UDP hlavička, a vypísaný zdrojový a cieľový port. Ďalej je získaná hlavička RIP správy, skontrolovaná jej veľkosť a vypísaný typ správy. Vetvenie odďeľuje spracovanie RIPv1 a RIPv2 správ. Spracovanie RIPv1 packetu pokračuje vypísaním verzie RIP, potom cyklus prejde všetky položky správy a vypíše ich. Pre RIPv2 funguje spracovanie podobne, vypíše verziu RIP a potom v cykle prechádza všetky položky správy. Pri prechádzaní položiek dochádza ku kontrole či prvá položka obsahuje informácie o autentifikácii (AFI tag == 65535). Pokiaľ položka obsauje textové heslo, tak je heslo vypísané. Pokiaľ obsahuje hashovaciu funkciu, sú vypísané jej parametre. Ďalej prebieha spracovanie a výpis informácii rovnako ako pre RIPv1.

process_ipv6()

Funkcia získa IP hlavičku a vypíše zdrojovú a cieľovú IPv6 adresu, potom UDP hlavičku a vypíše zdrojový a cieľový port. Následne získa hlavičku RIP správy, skontroluje jej veľkosť a vypíše typ správy a verziu RIP. Po získaní tela RIP správy cyklus prechádza všetky položky správy a vypisuje ich.

[1] https://www.wireshark.org/

3.2 Podvrhávač falošných RIPng Response správ

Po spustení aplikácie prebehne kontrola počtu argumentov a následne spracovanie zadaných arametrov pomocou funkcie `getopt()` [3]. Pri spracovaní argumentu `-r` očakávajúceho vstup vo formáte IPv6/prefix, je kontrolovaná validita IPv6 adresy pomocou funkcie `check_addr()`. Po spracovaní zadaných argumentov je skontrolované či boli zadané větky povinné parametre a či sú zadané číselné argumenty v povolenom rozsahu. Pri nezadaní nepovinných argumentov sú použité implicitné hodnoty (next hop adresa = ::, metrika = 1, router tag = 0). Po úspešnej kontrole je vytvorený UDP socket, ktorému je následne priradený port 521 pomocou funkcie `bind()` a rozhranie zadané užívateľom. Ďalej je socket nastavený pre použitie multicastu, a hop-count packetu odoslaného cez socket nastavený na 255. Pokiaľ vytvorenie a nastavenie socketu prebehlo správne, sú vytvorené a naplnené RIP štruktúry. Najprv hlavička, potom položka obsahujúca next-hop adresu a položka obsahujúca adresu siete. Po naplnení sú štruktúry nakopírované do bufferu znakov funkciou `memcpy()`. Po vytvorení packetu, je packet odoslaný funkciou `sendto()`. Pokiaľ odoslanie prebehlo v poriadku program vypíše hlášku oznamujúcu úspech, v prípade akejkoľvek chyby počas behu programu je vypísaná chybová hláška a program je ukončený s príslušným chybovým kódom.

check_addr()

Kód funckie bol prebratý zo stackoverflow.com[5]. Funkcia kontroluje či pole znakov prijaté ako argument odpovedá špecifikácii IPv6. V prípade IPv6 vracia 0, inak vypíše chybu a vráti 1.

3.3 Chybové kódy

10 (ERR_ARGS) chyba vstupných argumentov
11 (ERR_IP) nesprávny formát zadanej IPv6 adresy
50 (ERR_PCAP) chybové ukončenie funkcie z knižnice libpcap

4 Použitie

4.1 RIP sniffer

Spustenie aplikácie - ./myripsniffer -i <rozhranie>. Ukončenie behu programu zaslaním signálu SIGINT.

4.2 Podvrhávač falošných RIP response správ

Spustenie aplikácie - ./myripresponse -i <rozhranie>-r <IPv6>/[16-128] -n <IPv6> -m [0-16] -t [0-65535]. Argument -i zadáva sieťové rozhranie na ktoré bude útočný packet odoslaný, -r zadáva adresu podvrhávanej siete, -n zadáva next-hop adresu pre podvrhávanú sieť, -m zadáva metriku a -t zadáva router tag. Parametre v {} sú nepovinné.

4.3 Použitie aplikácií pre podvrhnutie routy

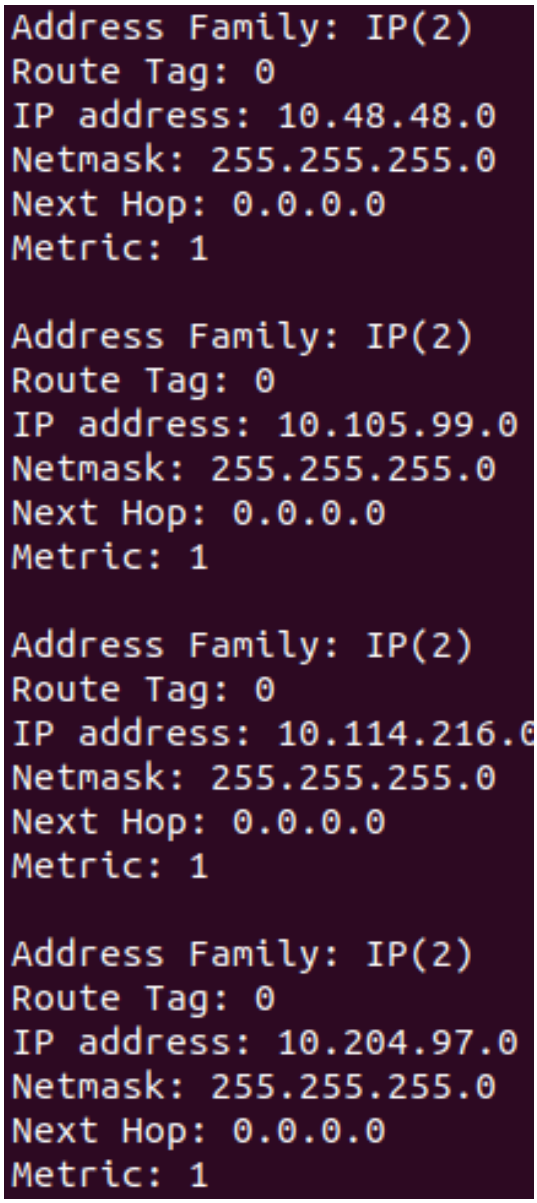
1. Spustenie RIP sniffer-u na sieťovom rozhraní vytvorenom virtuálnym smerovačom - `vboxnet0`. Je nutné spustiť aplikáciu s administrátorskými právami, inak dojde k chybe pri otváraní rozhrania.

```
$ sudo ./myripsniffer -i vboxnet0
```

2. Zo správ je možné zistiť autentifikačné heslo pre RIPv2 - „ISA>27c12007ff2“.

```
Authentication: Simple Password(2)
Password: ISA>27c12007ff2
Address Family: IP(2)
Route Tag: 0
IP address: 10.48.48.0
Netmask: 255.255.255.0
Next Hop: 0.0.0.0
Metric: 1
```

3. Pre IPv4 smerovanie boli odchytené 4 routy:
   (a) `10.48.48.0`
   (b) `10.105.99.0`
   (c) `10.114.216.0`
   (d) `10.204.97.0`



4. Pre IPv6 smerovanie bolo odchytených 5 rout:

   (a) `fd00::`

   (b) `fd00:d3:2d78::`

   (c) `fd00:fc:2772::`

   (d) `fd00:804:66::`

   (e) `fd00:900:13b0::`

```
IPv6 Prefix: fd00::                    Route Tag: 0x0000        Prefix Length: 64          Metric: 1
IPv6 Prefix: fd00:d3:2d78::            Route Tag: 0x0000        Prefix Length: 64          Metric: 1
IPv6 Prefix: fd00:fc:2772::            Route Tag: 0x0000        Prefix Length: 64          Metric: 1
IPv6 Prefix: fd00:804:66::             Route Tag: 0x0000        Prefix Length: 64          Metric: 1
IPv6 Prefix: fd00:900:13b0::           Route Tag: 0x0000        Prefix Length: 64          Metric: 1
```

5. Podvrhnutie routy `2001:db8:0:abcd::/64` virtuálnemu smerovaču a správa oznamujúca úspešné odoslanie packetu.

```
$ sudo ./myripresponse -i vboxnet0 -r 2001:db8:0:abcd::/64
**************************
Fake RIPng response tool
Response message parameters:
*Interface:  vboxnet0
*Network IP: 2001:db8:0:abcd::/64
*Next-hop:   ::
*Metric:     1
*Router tag: 0
**Success ! RIPng response message successfully sent**
```

6. Kontrola smerovacej tabuľky na virtuálnom smerovači. Routa bola úspešne pridaná, ako ukazuje záznam v smerovacej tabuľke - `R>* 2001:db8:0:abcd::/64 via ...`

```
Routing> show ipv6 route
Codes: K - kernel route, C - connected, S - static, R - RIPng, O - OSPFv3,
       I - ISIS, B - BGP, * - FIB route.

K>* ::/96 via ::1, lo0, rej
C>* ::1/128 is directly connected, lo0
K>* ::ffff:0.0.0.0/96 via ::1, lo0, rej
R>* 2001:db8:0:abcd::/64 [120/2] via fe80::800:27ff:fe00:0, em0, 00:02:06
C>* fd00::/64 is directly connected, em0
C>* fd00:d3:2d78::/64 is directly connected, lo0
C>* fd00:fc:2772::/64 is directly connected, lo0
C>* fd00:804:66::/64 is directly connected, lo0
C>* fd00:900:13b0::/64 is directly connected, lo0
K>* fe80::/10 via ::1, lo0, rej
C * fe80::/64 is directly connected, lo0
C>* fe80::/64 is directly connected, em0
K>* ff02::/16 via ::1, lo0, rej
```

Literatúra

[1] sniffex.c, Sniffer example of TCP/IP packet capture using libpcap, (Online, 18.11.2018)
https://www.tcpdump.org/sniffex.c

[2] Routing Information Protocol (RIP) Parameters, (Online, 26.10.2018)
https://www.iana.org/assignments/rip-types/rip-types.xhtml#rip-types-4

[3] Example of Parsing Arguments with getopt, (Online, 17.11.2018)
https://www.gnu.org/software/libc/manual/html_node/Example-of-Getopt.html

[4] RIPng Knowledge Base, (Online, 17.11.2018)
https://sites.google.com/site/amitsciscozone/home/important-tips/ipv6/ripng

[5] Tell whether a text string is an IPv6 address or IPv4 address using standard C sockets API, (Online, 10.10.2018)
https://stackoverflow.com/questions/3736335/tell-whether-a-text-string-is-an-ipv6-addr

[6] RFC1058 - RIP version 1, (Online, 10.10.2018)
https://tools.ietf.org/html/rfc1058

[7] RFC2453 - RIP version 2, (Online, 10.10.2018)
https://tools.ietf.org/html/rfc2453

[8] RFC2080 - RIPng for IPv6, (Online, 10.10.2018)
https://tools.ietf.org/html/rfc2080

[9] libpcap website, (Online, 12.10.2018)
http://www.tcpdump.org/